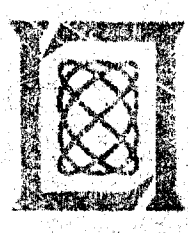
Multi-Dimensional Signal-Processing Research Program

31 March 1981

Prepared for the Department of the Air Force
and the Department of the Navy under Contract F19628-80-C-0002 by

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Lexington, Massachusetts



The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology, with the support of the Department of the Air Force under Contract F19628-80-C-0002. A part of this support was provided by the Rome Air Development Center.

The views and conclusions contained in this document are those of the contractor and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the United States Government.

The Public Affairs Office has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

Raymond L. Loiselle

Raymond L. Loiselle, Lt. Col., USAF
Chief, ESD Lincoln Laboratory Project Office

Non-Lincoln Recipients

PLEASE DO NOT RETURN

Permission is given to destroy this document when it is no longer needed.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LINCOLN LABORATORY

MULTI-DIMENSIONAL SIGNAL-PROCESSING RESEARCH PROGRAM

SEMIANNUAL TECHNICAL SUMMARY REPORT TO THE ROME AIR DEVELOPMENT CENTER

1 OCTOBER 1980 - 31 MARCH 1981

ISSUED 9 OCTOBER 1981

Original contains color plates: All DTIC reproductions will be in black and white

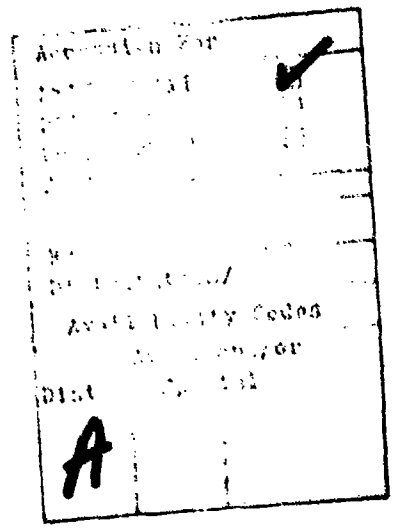
LEXINGTON

i/ii

MASSACHUSETTS

ABSTRACT

This Semiannual Technical Summary covers the period 1 October 1980 through 31 March 1981. It describes the significant results of the Lincoln Laboratory Multi-Dimensional Signal-Processing Research Program, sponsored by the Rome Air Development Center, in the areas of image segmentation and classification, adaptive contrast enhancement, and iterative implementations for multi-dimensional digital filters.



CONTENTS

MULTI-DIMENSIONAL SIGNAL-PROCESSING RESEARCH PROGRAM

1. INTRODUCTION AND SUMMARY

The Lincoln Laboratory Multi-Dimensional Signal-Processing Research Program was initiated in FY 80 as a research effort directed toward the development and understanding of the theory of digital processing of multi-dimensional signals and its application to real-time image processing and analysis. A specific long-range application is the automated processing of aerial reconnaissance imagery. Current research projects which support this long-range goal are image modeling for segmentation and classification, techniques for adaptive contrast enhancement, iterative implementation of multi-dimensional digital filters, and multiprocessor architectures for implementing image processing algorithms. Results in these research areas over the past six months are described in this Semiannual Technical Summary.

In the area of image segmentation and classification, we have been developing a hierarchical segmentation scheme for processing images with several region classes. This approach appears to offer improvements over a direct multiclass segmentation. Examples are given in Sec. 2.

Adaptive contrast enhancement techniques have proved useful in several areas. We have used adaptive contrast enhancement as a preprocessor for image segmentation based on texture rather than gray level. We have also applied these techniques to aerial images degraded by light cloud cover and haze. The primary effects of this type of degradation are a reduction in contrast and an increase in the local average intensity of the image. In Sec. 3, we show examples of the improvement which the adaptive contrast enhancement techniques afford on images suffering from this type of degradation.

In Secs. 4 and 5 we discuss two aspects of the iterative implementation of multi-dimensional digital filters. This first concerns spatial truncation effects which occur during the iteration because the image frame buffer has a finite storage capability. The errors caused by this truncation are closely-

related to the solution of a boundary value problem with boundary conditions specified on the frame edges. These errors can be eliminated by including the boundary conditions in the spatially truncated iteration.

Section 5 discusses potential multiprocessor architectures for realizing multi-dimensional signal-processing operations such as those needed in the iterative implementation. An underlying problem is the efficient partitioning of multi-dimensional signal-processing problems among the several processors in a multiprocessor architecture.

2. IMAGE SEGMENTATION

In the previous Semiannual Technical Summary Report,[1] we described a segmentation procedure based on linear filtering methods to model texture in local regions of an image and a Markov random field to model region transitions within the image. We further showed application of this method to images containing two region types. The segmentation algorithm described in Ref. 1 is also applicable to cases where there are more than two types of regions in the image. The conditions for the segmentation are as follows.

Assign a pixel with coordinates (n, m) to class i where i is the class for which

$$\frac{E_i^2(n,m)}{\sigma_i^2} + \ln \sigma_i^2 - 2 \ln \Pr\left[i \mid S_{n,m}\right] \qquad (1)$$

is minimum. In the above expression, E_i is the error in linear prediction of a pixel from a surrounding set of pixels, σ_i^2 is the variance of the prediction error, and $\Pr\left[i \mid S_{n,m}\right]$ is the probability that the given pixel belongs to class i given the classes of a surrounding set of pixels. (See Refs. 1 and 2 for more details.) During the current reporting period, we have been applying the multiclass form of the algorithm to aerial photographic data and experimenting with strategies for the segmentation of multiclass images. Our results indicate that it is often better not to attempt to segment an image into a large number of classes all at once. Instead, one can first segment the image into a few categories and then perform additional segmentation within each category. We

refer to this approach as a layered segmentation strategy. Several examples of using this layered approach are described in the following paragraphs.

In Ref. 1 we stated how an image which had been segmented into tree and field regions could be further segmented into regions containing only large or only small trees. Figure 1 compares results of this layered segmentation approach to a nonlayered approach for this image. The image in Fig. 1 (left) was first segmented into the broad categories of trees and fields. Within the tree category we further segmented the image into large and small tree regions; in the field category we segmented the image into two different field types. The region boundaries arising from this layered approach are shown in color. (Note that the small field in the lower left-hand corner has the same texture as the field adjacent to it and thus is not recognized as a separate region by the segmentation algorithm.) Figure 1 (right) shows the results of segmenting the image into the four regions in a single step. The regions identified by the algorithm are coded in different colors. Figure 1 shows that the layered segmentation strategy produced a more accurate result. Although both approaches gave quite accurate estimates of the boundary between the two field types and the boundary between the trees and field, the layered approach was able to make more subtle distinctions between the small and large trees. Both segmentation strategies had some problems near the boundaries of the image due to the discontinuities arising there but the problems were more apparent in the spot near the left edge of the image in Fig. 1 (right). The results of this comparison are not really surprising since in a four-category segmentation there is more room for error in the classification of the individual pixels than in successive two-category segmentations. A few points in the small tree region classified as "field" during the initial classification phase of the algorithm can propagate their effects through the iteration employed in the algorithm and lead to a final result that is incorrect. On the other hand, a layered approach may produce a large error in the initial categories that cannot be corrected at successive levels of segmentation. Thus, one can expect that a layered approach will produce best results where there is clear separation between the initial categories and subtle distinctions between the classes within those initial categories.

Fig. 1. Comparison of a layered segmentation (left) to a 4-class segmentation (right) of a rural scene.

109689-S

Another approach to layered segmentation is to first segment the image into regions representing various textures and a further category representing constant tones (no texture). Within the nontextured regions we perform a segmentation based on gray level. The results are then combined to produce the final segmentation of the image. This technique is particularly useful for the following reason. In segmenting an image for texture it is desirable to remove large overall variations in tonality and render the image an overall middle gray.[1,2] This eliminates artifacts in the segmentation due to variations in illumination, shadows, and so forth. However, preprocessing in this manner removes tonal features in nontextured regions that may be important for a complete and meaningful segmentation of the image. A meaningful result can be obtained by using a layered approach where a texture-oriented segmentation of the preprocessed image is followed by a gray-tone segmentation of the unprocessed image.

Figure 2 illustrates this approach for a scene containing trees, water, grass or fields, and a feature (apparently a road or bridge) passing over a small stream. Figure 2 (upper left) shows the original image and Fig. 2 (upper right) shows the preprocessed version. Note that in the preprocessed image the bright white area of the bridge has approximately the same tonal value as the dark shadow under the bridge. Figure 2 (lower left) shows the result of segmenting the original image into regions with three tonal values (dark, shown in red, middle grays, shown in green, and light, shown in blue). Figure 2 (lower right) shows the result of segmenting the preprocessed image into textures representing the trees (yellow), the grass and field areas (purple), and the nontextured regions such as water, the bridge, and the shadows (light blue). Figure 3 shows the original image with segmentation boundaries overlaid (between tree and field regions and between trees and water) in solid lines. In addition, the boundary obtained by overlaying the tonal separation of Fig. 2 (lower left) on the nontextured areas of Fig. 2 (lower right) is shown in gray. This permits us to obtain a more accurate representation of the bridge segment.

As another example of this layered approach, consider the portion of the scene shown in the white box in Fig. 4(a) and enlarged in Fig. 4(b). The enlarged scene is 256 x 256 pixels in size and represents the full resolution of

Fig. 2. Layered approach to segmenting a rural scene with textured and nontextured areas. Original image (upper left), preprocessed image (upper right), texture segmentation (lower left), and gray-level segmentation (lower right).

109690-S

01/6

Fig. 3. Rural scene with boundaries resulting from layered segmentation superimposed.

(a) (b)

Fig. 4. Enlarged portion of a river scene used for layered segmentation experiment.

the data. (The photograph from which the digital data were taken was scanned at a resolution of 40 μm.) Figure 5 shows the original image and the result after segmenting the image into textured regions (trees) and nontextured regions and applying three-level gray-tone segmentation within the nontextured regions. Note that the dark shadows that the trees cast on the water are easily separated from both the trees and the water using this procedure. The isolation of shadowed areas may be important in deducing the height of objects in a scene.

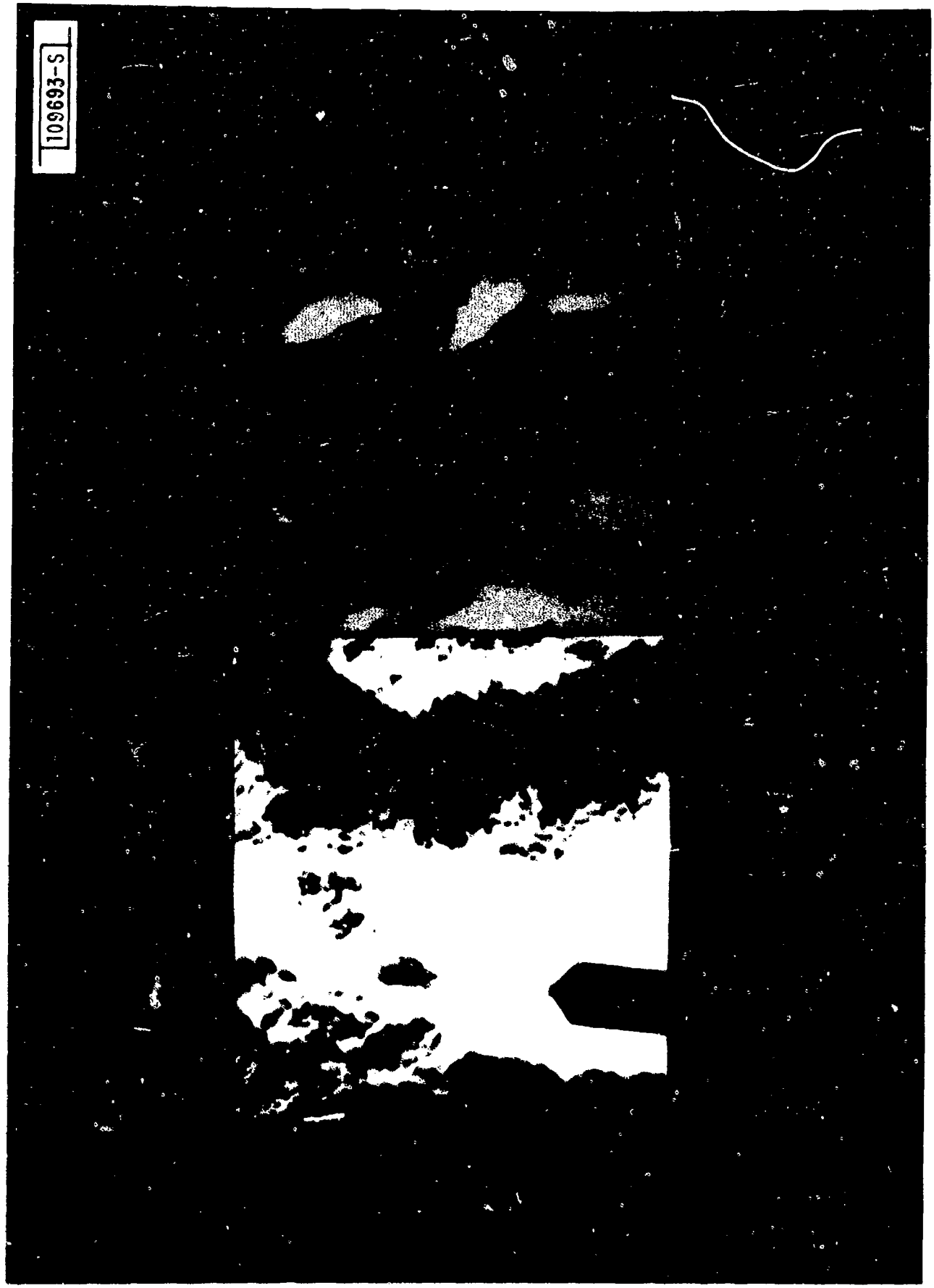
A final example of segmenting textured from nontextured regions is shown in Fig. 6. The original scene of roadways connecting missile sites is shown in the (a) part of the figure. The two-category segmented result is shown in the (b) portion. The segmentation of this scene is a difficult problem because the roads tend to bleed into the surrounding terrain in many places and the presence of features on the roads (such as the objects in the corner of the parking lot) tend to produce something like a small textured area. Nevertheless, a fairly good approximation to the roadways is obtained. Ideally we would wish to follow this texture segmentation with a gray-level segmentation to highlight features in the road. However, these features, because of their size, have tended to be confused with the texture and thus many do not appear in the region designated by the algorithm as roadways. This is a problem that requires additional work. The features in the roadway can perhaps be detected by scrutinizing the error residuals in the segmentation algorithms or possibly by employing more texture classes.

Our current research is directed toward resolving some of the problems described above and to developing a method for automatic selection of the training data (by the segmentation algorithm). The latter would allow the segmentation algorithm to function in a nonsupervised mode and eliminate the need for human interaction to define sets of training data.

3. TECHNIQUES FOR REMOVING DEGRADATIONS CAUSED BY LIGHT CLOUD COVER

In this section, we will discuss the use of homomorphic filtering to expose objects beneath light cloud cover. In particular, a deterministic model of

Fig. 5. Segmentation of the river scene in Fig. 4(b).


109693-S

(a)

(b)

Fig. 6. Two-class segmentation of an aerial photograph of a missile site.

imaging above cloud cover motivates an approach which utilizes an adaptive homomorphic filter. This space-varying filter is parameterized by the local mean level which reflects the degree of the local degradation.

The approach we take is distinctly different from other homomorphic filtering procedures for image enhancement or restoration. It differs from the work of Mitchell and Delp,[3] which recovers images degraded by light cloud cover, since their scheme relies on a stochastic image. It differs from the deterministic approach of Oppenheim et al.[4] and Gilkes[5] for image enhancement of cloudless images, since again theirs are nonadaptive procedures. In essence, it is closer to the adaptive approaches of Peli and Lim[6] and Gilkes[5] where an adaptive filter is parameterized by the local deterministic characteristics of the data.

One long-space model of a cloudy image is described in Ref. 3. Specifically, this model of a cloudy image is stochastic and is a product of the cloud transmission function and a function of the ideal image. Our cloudy image model is deterministic and applies on a short-space basis. It assumes that the logarithm of the image can be divided into two approximately disjoint spectral bands: the cloud spectrum occupying low frequencies and the image spectrum occupying high frequencies. Although the image contains some low-frequency information, we shall assume it is not significant in exposing object shapes under light cloud cover. In addition, we depart from a typical assumption that the desired image itself can be modeled as the product of illumination, a low-frequency component, and reflectivity, a high-frequency component. Rather, we view the illumination component in the same way as we view the reflectance component, i.e., as having an important high-frequency component due to the interaction of light and ground objects.[7] Furthermore, we assume this high-frequency component is approximately disjoint from the cloud transfer function which has a low-pass characteristic.

Before proceeding with the development of our new methods and comparisons, we review some important ideas and formulate a framework for our investigations.

3.1 Modeling the Imaging Process

A number of approaches to modeling the imaging process have been presented in the literature. A more difficult problem is to model the imaging process with light cloud cover. In this section, we first review one approach to modeling undegraded images. Two different viewpoints are presented which rely on an illumination-reflectivity model. We then enter a stochastic framework in which a model of images degraded by light cloud cover will be discussed. Finally, we present our own deterministic interpretation of cloudy images.

In the formation of images, the illumination and reflectivity are combined by a multiplication law:[4]

$$I(n, m) = i(n, m) \cdot r(n, m) \tag{2}$$

where $i(n, m)$ and $r(n, m)$ are the illuminance and reflectance components, respectively. One assumption is that the illumination varies slowly (i.e., it contains mainly low-frequency components) and the reflectance is sometimes dynamic and sometimes static and therefore may be regarded as containing mainly high-frequency components. The logarithm operation separates the multiplicative signal into two additive components:

$$\log [I(n, m)] = \log [i(n, m)] + \log [r(n, m)] \quad . \tag{3}$$

If a linear amplifier or attenuator with gain α follows the log, the image output is given by

$$I'(n, m) = [I(n, m)]^{\alpha} \quad . \tag{4}$$

When $\alpha < 1$, we get a washed-out image and when $\alpha > 1$, the image is sharpened but might be saturated. If we want to reduce the dynamic range, which is contributed mostly by the illumination, and increase the sharpness, which is contributed mostly by the reflectance, we should choose α as a function of frequency, i.e., $\alpha < 1$ for high frequencies as in Fig. 7. This is the essence of the homomorphic filtering operation.

Since the illumination is not exclusively low frequency and the reflectance consists of both high- and low-frequency components, the output of the homomorphic filtering operation has some artifacts. It seems that the degree to

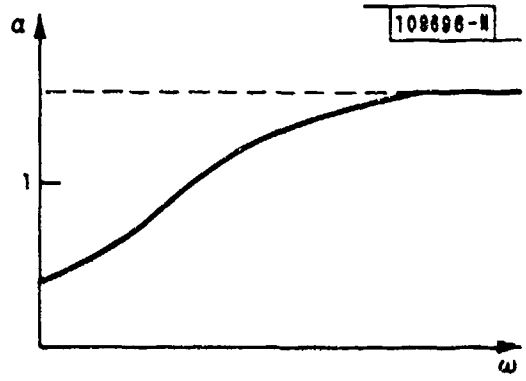


Fig. 7. Variability of gain α as a function of frequency.

which those artifacts are visible is strongly dependent on the way α changes from $\alpha < 1$ to $\alpha > 1$. As pointed out by Schreiber,[7] homomorphic filtering can be replaced by a linear filter followed by a power law for low-contrast images. Schreiber hypothesizes that in natural scenes, the illumination function is as essential to perception as the reflectivity of the object. Both have a broad spectrum, generally with more power at the low spatial frequencies. The reflectance derives its low-frequency component from the presence of large patches of relatively constant value. The illumination derives its high-frequency component from interaction between the edges and surfaces of objects, which are at many different angles, and the incident light. Consequently, any attempt to separate these components by homomorphic filtering will be limited in its success. Instead, adaptive control of the low and high components of the original signal is preferable for enhancement.

Now let us examine one model of images degraded by cloud cover.[3] In this model the ground reflection passes through the clouds. The energy collected above the cloud consists of two components: ground information and scattering from the clouds. Figure 8 is a simplified model of the imaging process. The energy recorded is given by:

$$s(n, m) = aLr(n, m)\, t(n, m) + L\,[1 - t(n, m)] \tag{5}$$

where

L = sun illumination

a = attenuation of the illumination in a downward direction (assumed constant)

$r(n, m)$ = reflectance of the ground

$t(n, m)$ = transmittance of the cloud (upward),

$0 \leqslant t(n, m) \leqslant 1$.

In order to recover the desired reflectance information, it was assumed that the transfer function of the cloud and the reflectance of the ground are

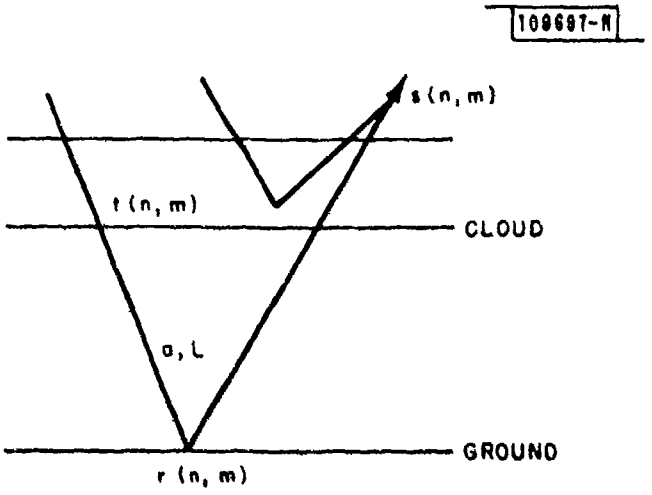


Fig. 8. A simplified model of imaging through clouds.

stochastic processes. It was also assumed that the transmittance of the cloud has relatively more energy in low spatial frequencies. From Eq. (5), we can obtain a multiplicative form of the "noise" and the "signal" given by:

$$L - s(n, m) = t(n, m)\,[L - aLr(n, m)] \quad . \tag{6}$$

By taking the log of Eq. (6), we obtain the sum

$$\log[L - s(n, m)] = \log[t(n, m)] + \log[L - aLr(n, m)] \tag{7}$$

or

$$P(n, m) = N(n, m) + M(n, m)$$

where

$$P(n, m) = \log[L - s(n, m)] = \text{"signal + noise"}$$
$$N(n, m) = \log[t(n, m)] = \text{"noise"}$$
$$M(n, m) = \log[L - aLr(n, m)] = \text{"signal"} \quad .$$

3.2 Filtering Based on the Image Model[3]

The recovery of the signal from Eq. (7) can be approached by the application of Wiener filtering. If we assume that the signal and the noise are uncorrelated, we can reduce the noise by filtering the given degraded image with the following optimal filter

$$H(\omega_1, \omega_2) = \frac{S_{\mu\mu}(\omega_1, \omega_2) + M_\mu \cdot M_N \cdot \delta(\omega_1, \omega_2)}{S_{\mu\mu}(\omega_1, \omega_2) + S_{NN}(\omega_1, \omega_2) + 2M_\mu \cdot M_N \delta(\omega_1, \omega_2)}$$

$$= \frac{S_{\mu p}(\omega_1, \omega_2)}{S_{pp}(\omega_1, \omega_2)} \tag{8}$$

where

M_μ is the mean value of the given signal M

M_N is the mean value of the given noise N.

In order to estimate S_{pp}, the power spectrum of the signal plus noise and S_{NN}, the power spectrum of the noise, the values L and t(n, m) need to be calculated. L was estimated as the highest value in the image and t(n, m) was estimated by the following:

$$\hat{t}(n, m) = \frac{L - S(n, m)}{L - G} \tag{9}$$

where G is a constant which approximates the typical ground reflection [i.e., the average of aLr(n, m)].

The power spectrum of the noise was estimated roughly by the magnitude squared of the Fourier transform of $\hat{t}(n, m)$. Since the transfer function of the cloud was considered to possess only low frequencies, the estimate of this power spectrum was low-pass filtered. The cross section of the resulting filter is given in Fig. 9.

The filter has a boost at $(\omega_1, \omega_2) = (0, 0)$ as a result of assuming a non-zero mean process and an attenuation at the low frequencies. Except for the value at (0, 0), the filter has basically the same functional form as the gain in the homomorphic filtering approach. It should be noticed that in homomorphic

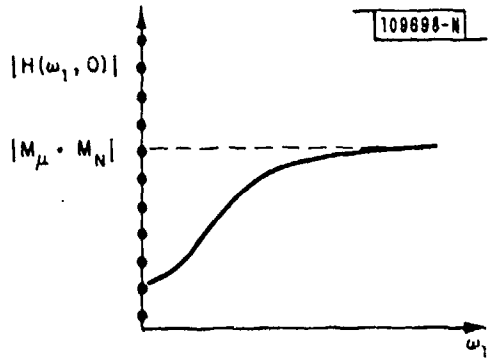


Fig. 9. Cross section of the 2-D Wiener filter.

filtering, it is possible to sharpen the edges while in the Wiener filter approach, it is not since $|H(\omega)| \leqslant 1$ for all $\omega \neq 0$.

In general, the noise spectrum density $S_{NN}(\omega_1, \omega_1)$ is not known accurately and consequently, the Wiener filter used to process the noisy image may be suboptimal. An alternative approach involves iteratively updating the estimate of the spectral density of the noise. Since an estimate of the reflectance function aLr(n, m) is used in determining G in Eq. (9), the improved estimate of aLr(n, m) obtained from the Wiener filter output can be used to update the estimate of the noise spectral density. The Wiener filter in Eq. (8) is then recomputed and the original cloudy image is again filtered. If we do this repeatedly, the resulting iterative process should converge to a better estimate than generated by the one-step process. In the space domain, the iterative cloud estimate is given by

$$\hat{t}_1(n, m) = \frac{L - s(n, m)}{L - G_0} \tag{10a}$$

where G_0 is a constant. From $\hat{t}_1(n, m)$, we get a better estimate of aLr(n, m) denoted by $G_1(n, m)$, and the iteration is continued as

$$\hat{t}_{k+1}(n, m) = \frac{L - S(n, m)}{L - G_k} \tag{10b}$$

where G_k is the k^{th} estimate of the function aLr(n, m).

We can introduce a modification of this iteration where, rather than filtering the original cloudy image on each iteration, we filter the updated signal estimate [i.e., filter $G_{k+1}(n, m)$ by using $G_k(n, m)$ to create the new signal estimate $G_{k+2}(n, m)$]. In the space domain, the iterative cloud estimate is given by

$$\hat{t}_{k+2}(n, m) = \frac{L - G_{k+1}}{L - G_k} \qquad k = 1, \ldots \quad . \tag{11a}$$

The initial conditions G_0 and G_1 are estimates as follows:

$$\hat{t}_1(n, m) = \frac{L - s(n, m)}{L - G_0} \tag{11b}$$

where G_0 is a constant and

$$\hat{t}_2(n, m) = \frac{L - s(n, m)}{L - G_1} \quad . \tag{11c}$$

This procedure appears to generate a better restoration.

3.3 Homomorphic Filtering

The assumption that images and clouds are stochastic processes is very artificial. A more appropriate approach invokes the assumption that the two are deterministic processes. In fact, we adopt Schreiber's approach[7] which assumes that both illumination and reflectance contain low and high components and that the high frequencies are more important for perception. At the same time, we assume that the transfer function of the cloud contains only low spatial frequencies. We now interpret Eq. (6) so that $t(n, m)$ is deterministic and contains mainly low frequencies and $L - aLr(n, m)$ is also deterministic and contains perceptually important high-frequency information. By using these assumptions, the image $aLr(n, m)$ can be restored by homomorphic filtering.

As discussed in Sec. 3.2, both the image illumination and reflectance contain low- and high-frequency components. In our application, however, the perceptually important high-frequency component of illumination is actually a blessing. That is, since light cloud cover is primarily low pass, it can be

homomorphically filtered without significantly disturbing the high-frequency component of the desired image's illumination and reflectance. Of course, the desired low-frequency components will be altered. Nevertheless, these changes should reflect themselves simply as illumination changes over large constant regions, thus not disturbing any significant information under the light cloud cover.

An implied assumption here is that the cloud spectrum contains predominantly low-frequency components, which motivates the long-space homomorphic filtering solution. In fact, this is similar to the assumptions made by Mitchell et al.[3] in a stochastic framework. However, the thickness of cloud cover can change over the extent of an image. Consequently, an adaptive approach should be better suited to restoring the desired image, i.e., an approach where the homomorphic filter changes with local cloud characteristics.

In earlier work,[6,8] we developed a locally adaptive contrast enhancement technique based on high-pass filtering. The local mean of the cloudy image was used to determine how much gain was applied to the high-frequency components of the image in a particular area. In this way the amount of contrast increase was adaptively adjusted across the entire image. The adaptive homomorphic filtering approach described here is similar in that measurements made on local image characteristics are used to determine the homomorphic filter characteristics.

The adaptive technique can be formulated as follows. Consider applying a short-space window to the noisy signal P(n, m) in Eq. (7):

$$P_{\ell,k}(n,m) = W_{\ell,k}(n,m) \cdot P(n,m) = W(n - \ell M/2,\ m - kM/2) \cdot P(n,m) \quad . \tag{12}$$

Let us assume the window takes on a pyramidal shape and is shifted over the data at intervals of half its length (i.e., M/2 where M × M is the extent of the window). Consequently, the window has the following desirable property:

$$\sum_{\ell} \sum_{k} W(n - \ell M/2,\ m - kM/2) = 1 \quad . \tag{13}$$

Applying an adaptive filter which operates on each segment $P_{\ell,k}(n,m)$, we obtain:

$$\hat{P}_{\ell,k}(n,m) = P_{\ell,k}(n,m) ** h_{\ell,k}(n,m,\Theta)$$

$$= [W(n - \ell M/2,\ m - kM/2)\ P(n,m)] ** h_{\ell,k}(n,m,\Theta) \qquad (14)$$

where the parameterized filter impulse response $h_{\ell,k}(n,m,\Theta)$ is a function of the vector Θ, which is a set of parameters dependent on the local cloud characteristics. (The double asterisk denotes 2-D spatial convolution.) In particular, Θ is a function of the DC level of the windowed signal which reflects the cloud density under the window. Further, the filter is a high pass where the shape and amplitude depend on Θ. Later we will elaborate on this design. For Eq. (7), the filtering process is then given by

$$\hat{P}_{\ell,k}(n,m) = \Big(W(n - \ell M/2,\ m - kM/2)\ \{\log[t(n,m)]$$

$$+ \log[L - aLr(n,m)]\}\Big) ** h_{\ell,k}(n,m,\Theta) \quad . \qquad (15)$$

We assume that the window is "sufficiently smooth" so that the low-pass nature of the noise term $\log[t(n,m)]$ and the high-pass nature of the signal component $\log[L - aLr(n,m)]$ are preserved after windowing. Consider the case where the noise and the signal are disjoint in frequency, and the filter $h_{\ell,k}(n,m,\Theta)$ is an ideal high-pass filter whose nonzero energy band matches that of the signal, then we can write

$$\hat{P}_{\ell,k}(n,m) = W(n - \ell M/2,\ m - kM/2)\ \log[L - aLr(n,m)] \quad . \qquad (16)$$

In practice, however, these assumptions do not hold exactly. Since the noise and signal are only approximately disjoint, the noise term will not be entirely removed, and since the filter is not an ideal high pass, the signal will not remain intact. Alternatively, where $h_{\ell,k}(n,m,\Theta)$ adaptively amplifies the high frequencies and attenuates the lows, the cloud will be only partially suppressed {while somewhat disturbing the low-frequency component of $\log[L - aLr(n,m)]$} and the high-frequency detail of the desired signal will be enhanced (while boosting any cloud energy in this region). The relation in Eq. (16) represents an approximation of the desired windowed signal along with the effects of any residual cloud noise. The parameters Θ, in $h_{\ell,k}(n,m,\Theta)$, which rely on the

local cloud density should be chosen so that the noise is suppressed as much as possible without altering the desired signal.

With these approximations in mind, our reconstruction procedure (which we refer to as overlap-add) can then, with the use of Eq. (13), be written as

$$P(n,m) = \sum_{\ell} \sum_{k} W(n - M\ell/2,\ m - Mk/2) \log[L - aLr(n,m)]$$

$$= \log[L - aLr(n,m)] \sum_{\ell} \sum_{k} W(n - M\ell/2,\ m - Mk/2)$$

$$= \log[L - aLr(n,m)] \quad . \tag{17}$$

This procedure is illustrated in Fig. 10. Finally, to obtain an estimate of the signal, we exponentiate P(n, m) to recover aLr(n, m).

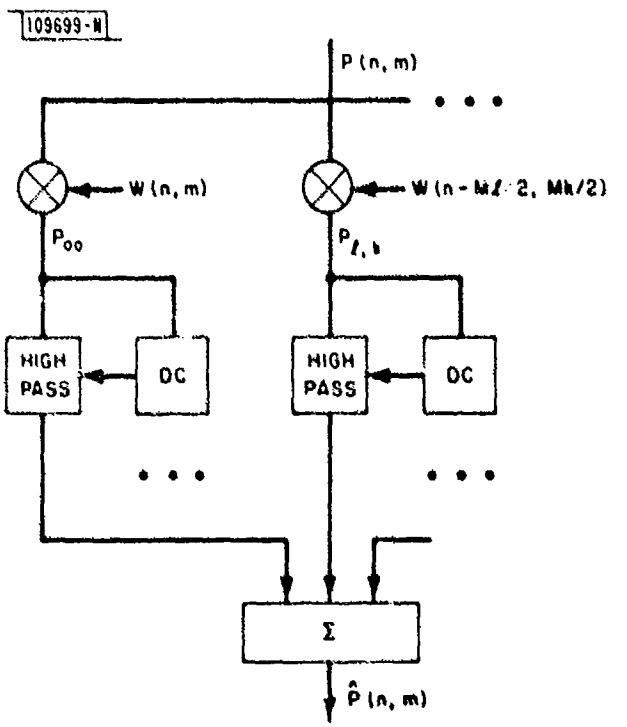


Fig. 10. Overlap-add technique for adaptive image filtering.

From the simple model of imaging above clouds, we have concluded that it is desirable to adaptively attenuate the low frequencies and to amplify the high frequencies of an image. A filter which has the desirable shape and is computationally efficient is a circularly symmetric Gaussian which is shifted to $\omega = \pi$. A cross section of this filter is given in Fig. 11 where $|H(0)| < 1$,

$$H(\omega_1, \omega_2) = A \cdot \exp\left[-\left(\frac{\omega_1 - \pi}{B^2}\right) - \left(\frac{\omega_2 - \pi}{B^2}\right)^2\right] + C \quad . \qquad (18)$$

The filter size was chosen to be fixed at 16×16. The filter parameters A, B, and C are functions of the value of the Fourier transform of the windowed data at $(\omega_1, \omega_2) = (0, 0)$ (i.e., the DC value of the pyramid-shaped window). The filter parameters are given for two extreme DC values: (A1, B1, C1) for a DC level of zero and (A2, B2, C2) for a DC level of 225. For a DC value that lies between these two extremes, the parameters are computed by a quadratic equation

$$Y = \frac{(Y_2 - Y_1) \cdot D^2}{255^2} + Y_1 \qquad \text{for } Y = A, B, C \qquad (19)$$

where D denotes the DC level and where Y_1 and Y_2 represent the two extreme values of each parameter.

It is desirable that for a low value of the DC level (i.e., little or no cloud cover), very little attenuation and amplification should be used and, further, the amplification should be applied to only the very high frequencies (i.e., B, the standard deviation of the Gaussian shape in Fig. 11, should be small). For a high DC level (i.e., thick clouds), the attenuation and amplification should be applied to a wider range of high frequencies (i.e., B should be larger). A quadratic interpolation for computing A, B, and C was used rather than a linear interpolation because a larger range of low level of luminance needs almost no processing.

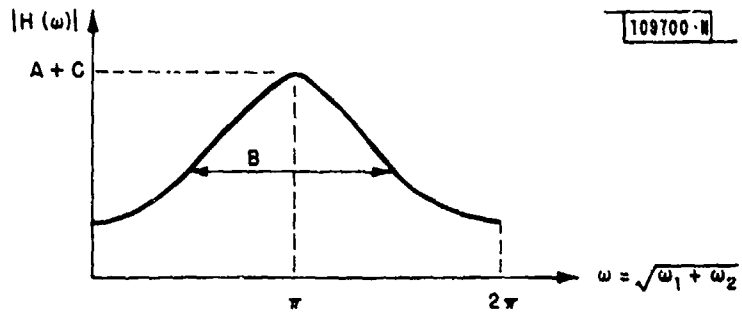


Fig. 11. Cross section of a Gaussian high-pass filter.

3.4 Experimental Results

An experiment was performed in order to compare the Wiener filtering approach to the homomorphic filtering approach for removing degradations caused by light cloud cover. An aerial reconnaissance image partially degraded by light cloud cover is shown in Fig. 12. Figure 13 is the result of the Mitchell and Delp[1] algorithm when the estimate of the typical ground reflection G was chosen to be a constant equal to 140. The processed image seems to contain only the high frequencies of the unprocessed image.

By using the iterative approach described by Eqs. (10a) and (10b) one can improve the estimate of the noise spectrum. Figure 14 contains the result after three iterations.

Figure 15 is the result after three iterations of improving the estimate of the noise and also reprocessing the processed image as in Eq. (11a). One can see that the second form of iteration is more desirable.

Long-space homomorphic filtering was examined with two different filters. Each has a Gaussian shape as described in Sec. 3.3 and a length of 256 × 256 (i.e., the length of the unprocessed image). The parameters of the first filter are A = 3.65, B = 320, and C = −2.15. The processed image is shown in Fig. 16. The second filter which sharpens and attenuates more than the first has the parameters A = 6.21, B = 320, and C = −4.21. The processed image is illustrated in Fig. 17. From these results, we can see that there is some

Fig. 12. Aerial photograph with some light cloud cover.

Fig. 13. Image processed by application of a global Wiener filter applied to log [L – s(n, m)]. (See Ref. 1.)

Fig. 14. Image processed with a global Wiener filter iterated three times to obtain improved estimate of noise power spectrum.

Fig. 15. Image processed according to Eq. (11) with three iterations.

Fig. 16. Image processed with a long-space homomorphic high-pass filter with parameters A = 3.65, B = 320, and C = −2.15.

Fig. 17. Image processed with a long-space homomorphic high-pass filter with parameters A = 6.21, B = 320, and C = −4.21.

sharpening but most of the cloud was not removed and, therefore, some objects were not exposed.

In the adaptive homomorphic filtering approach, a pyramidal window of size 16 × 16 was chosen and the parameters ranges are given in Table I. The filter shapes are shown in Fig. 18, and the processed image is illustrated in Fig. 19.

TABLE I

RANGE OF PARAMETERS FOR THE ADAPTIVE HOMOMORPHIC FILTERING

DC	0	255
A	0.2	1.82
B	5.0	20.0
C	0.9	−0.32

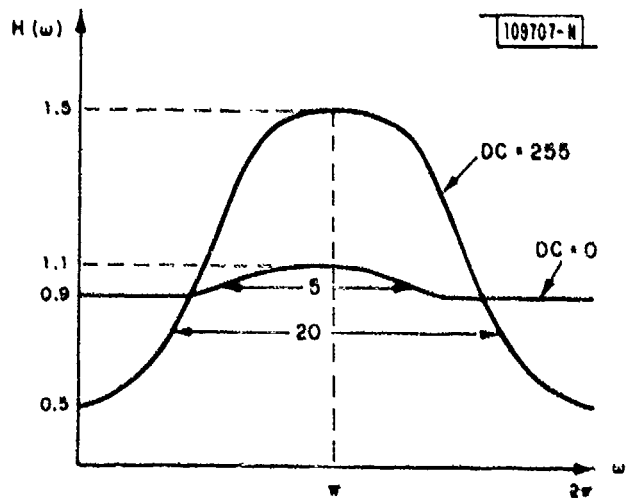


Fig. 18. Range of filter shapes for adaptive homomorphic system.

Fig. 19. Image processed with adaptive homomorphic filter.

We see from our results that the most desirable restoration, i.e., maximum suppression of the cloud and enhancement of the objects, was achieved by adaptive homomorphic filtering. We are beginning to explore implementation strategies for this algorithm to improve its computational efficiency.

4. ERRORS CAUSED BY TRUNCATION EFFECTS IN THE ITERATIVE IMPLEMENTATION OF MULTI-DIMENSIONAL DIGITAL FILTERS

When an iteration is used to compute successively better approximations to the output of a digital filter, the effective impulse response grows in extent at each iteration. For the case where the frame storage buffer has a fixed size, the extent of the successive output estimates is limited by the buffer size, resulting in spatial truncation errors which affect successive approximations to the desired output signal. In this section, we shall discuss our preliminary findings on truncation errors and their relation to boundary conditions. We begin by giving a brief review of the iterative implementation.

4.1 Brief Review of the Iterative Implementation

The iterative filter possesses an interesting structure to implement, since it involves FIR (finite-extent impulse response) filtering operations as well as the feedback of buffered output image frames. Because of this frame feedback, the iterative filter can be used to approximate the output of a 2-D symmetric rational transfer function. Consider the filter frequency response of the form

$$H(\omega_1, \omega_2) = \frac{A(\omega_1, \omega_2)}{B(\omega_1, \omega_2)} \tag{20}$$

where $A(\omega_1, \omega_2)$ and $B(\omega_1, \omega_2)$ are 2-D trigonometric polynomials with coefficients $a(n_1, n_2)$ and $b(n_1, n_2)$, respectively. With appropriate normalization, we can write

$$B(\omega_1, \omega_2) = 1 - C(\omega_1, \omega_2) \tag{21}$$

where $C(\omega_1, \omega_2)$ is another trigonometric polynomial with coefficients $c(n_1, n_2)$. Let us filter an input image $x(n_1, n_2)$ to obtain an output image $y(n_1, n_2)$. Then $x(n_1, n_2)$ and $y(n_1, n_2)$ satisfy the implicit relation

$$y(n_1, n_2) = a(n_1, n_2) ** x(n_1, n_2) + c(n_1, n_2) ** y(n_1, n_2) \qquad (22)$$

where the double asterisk denotes 2-D convolution. We can use this relation iteratively to generate successively better approximations to $y(n_1, n_2)$. Letting i denote the iteration number, we can write

$$y_i(n_1, n_2) = a(n_1, n_2) ** x_i(n_1, n_2) + c(n_1, n_2) ** y_{i-1}(n_1, n_2) \qquad (23a)$$

or in the frequency domain

$$Y_i(\omega_1, \omega_2) = A(\omega_1, \omega_2)\, X_i(\omega_1, \omega_2) + C(\omega_1, \omega_2)\, Y_{i-1}(\omega_1, \omega_2) \quad . \qquad (23b)$$

The subscript i on the input image $x_i(n_1, n_2)$ is included to indicate that the iterative implementation may be generalized to accept sequences of image frames as its input. Similarly, the image sequence $y_i(n_1, n_2)$ may be regarded as a succession of image frames. At this point, it is natural to consider the iteration index i as a time variable.

Figure 20 shows a simple block diagram for the iterative implementation which highlights the basic operations. There are two FIR filtering operations, denoted by the boxes labeled $A(\omega_1, \omega_2)$ and $C(\omega_1, \omega_2)$, a summing node, and a frame storage buffer.

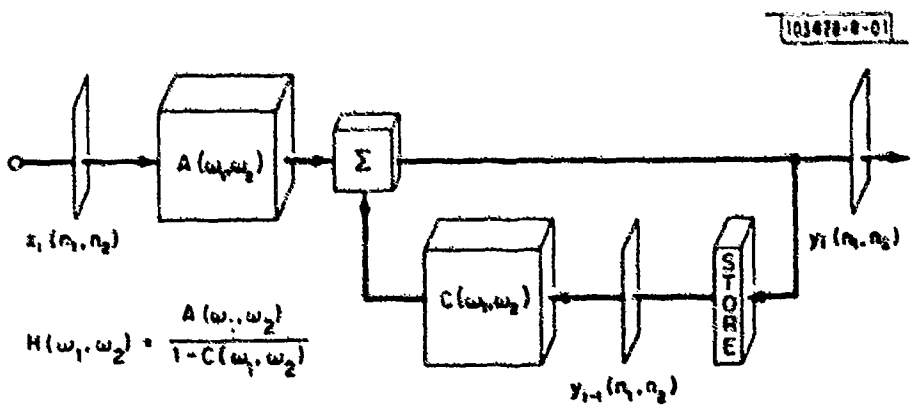


Fig. 20. Block diagram of the 2-D iterative filter implementation.

4.2 Characterization of the Truncation Error

The operation of truncation is particularly important since we cannot deal in practice with infinitely long sequences. Because the output of a rational filter has infinite extent in general, errors will be introduced by repeated truncation and may become intolerably large over the limited domain of the output signal. The error over this limited region can be viewed as the solution to a homogeneous partial difference equation with specified boundary conditions. In particular, the error is linearly dependent on values of the ideal solution along the boundary of the region remaining after truncation.

The iteration (23a) can be written in operator notation as

$$y_i(n_1, n_2) = Fy_{i-1}(n_1, n_2) \tag{24}$$

where F is an operator of the form

$$Fy(n_1, n_2) = a(n_1, n_2) ** x(n_1, n_2) + c(n_1, n_2) ** y(n_1, n_2) \quad . \tag{25}$$

Under the appropriate assumptions, there exists a unique solution $y(n_1, n_2) = Fy(n_1, n_2)$. When the size of the frame buffer is exceeded, the 2-D signal $y_i(n_1, n_2)$ must be truncated in extent. This introduces a truncation operator T into the iteration, giving Eq. (24) the form

$$y_i(n_1, n_2) = TFy_{i-1}(n_1, n_2) \tag{26}$$

where

$$Ty(n_1, n_2) = \begin{cases} y(n_1, n_2) & \text{for } (n_1, n_2) \in I \\ 0 & \text{for } (n_1, n_2) \notin I \end{cases} \quad . \tag{27}$$

(The region I can be thought of as representing the extent of the frame buffer.) The iteration (26) also can be shown to have a unique solution $\hat{y}(n_1, n_2) = TF\hat{y}(n_1, n_2)$. In general, the solution to Eq. (24), $y(n_1, n_2)$, will not be equal to $\hat{y}(n_1, n_2)$. The difference between these two signals is attributable to the truncation effects. Let us define the truncation error as

$$e(n_1, n_2) = \hat{y}(n_1, n_2) - y(n_1, n_2) \quad . \tag{28}$$

It can be shown that the error signal $e(n_1, n_2)$ satisfies an iteration which corresponds to driving Eq. (23a) with no input $[x_i(n_1, n_2) = 0]$ but imposing boundary conditions around the edges of the region I. Let us consider a region I_b which surrounds the region I. The extent of I_b depends on the extent of impulse response $c(n_1, n_2)$ in Eq. (23a). We can now define the boundary conditions $bnd(n_1, n_2)$ to be

$$bnd(n_1, n_2) = \begin{cases} y(n_1, n_2) & \text{for } (n_1, n_2) \in I_b \\ 0 & \text{elsewhere} \end{cases} . \tag{29}$$

Then the error signal $e(n_1, n_2)$ satisfies the iteration

$$e_i(n_1, n_2) = T[c(n_1, n_2) ** e_{i-1}(n_1, n_2)] + bnd(n_1, n_2)$$
$$\text{for } (n_1, n_2) \in I + I_b \quad . \tag{30}$$

The energy in the error signal can be shown to be proportional to the energy in the boundary condition signal $bnd(n_1, n_2)$. Thus, if the size of the frame buffer (and hence the extent of the region I) is large enough, one would expect that the correct solution $y(n_1, n_2)$ [and hence the boundary condition signal $bnd(n_1, n_2)$] would be small in the region I_b. In this case, the error signal $e(n_1, n_2)$ also will be small, and $\hat{y}(n_1, n_2)$ will be a good approximation to the desired output signal $y(n_1, n_2)$. Furthermore, the error can be eliminated by including the boundary condition information $bnd(n_1, n_2)$ in the spatially truncated iteration, giving it the form

$$y_i(n_1, n_2) = TFy_{i-1}(n_1, n_2) + bnd(n_1, n_2)$$
$$\text{for } (n_1, n_2) \in I + I_b \quad . \tag{31}$$

5. POTENTIAL ARCHITECTURES FOR THE ITERATIVE IMPLEMENTATION OF MULTI-DIMENSIONAL DIGITAL FILTERS

In this section, we shall examine some architectures which support the iterative implementation of multi-dimensional digital filters. We begin by

discussing possible data-handling approaches and their implications with respect to architectures. Later, we shall take a "software" point of view and discuss the problem of partitioning signal-processing operations among several processors in a multiprocessor architecture.

Referring back to Fig. 20, we see that the iterative implementation consists of two FIR filtering operations, denoted by the boxes labeled $A(\omega_1, \omega_2)$ and $C(\omega_1, \omega_2)$, a summing node, and a frame storage buffer. Consequently, much of the discussion of processing architectures will be directed toward the efficient implementation of multi-dimensional FIR filters.

5.1 Image Flow Options

There are three popular methods for describing how a sequence of images, each composed of many picture elements (pixels), may flow through a signal-processing architecture. The iterative implementation as diagrammed in Fig. 20 suggests a "frame-by-frame" image flow which might result naturally from a lens imaging system; all the image pixels from a given frame are available simultaneously. One of the major problems with this image flow is providing the large number of parallel data paths necessary to shuttle entire images around the processing structure. Conversely, one advantage is the relatively long available data transfer time, typically 1/30 s. Consequently, there is flexibility in trading data transfer time for the number of data paths using multiplexors and demultiplexors.

At the other extreme of the image-data-flow spectrum we have the "sequential pixel" flow. All the pixels in an image are transferred one after another (multiplexed), followed by the pixels of the next image frame, and so on. The processing structure sees a continuing stream of pixel values. This serial flow of image data can result from sensors such as the flying spot scanner, where a single detector is swept across an image in a raster scan. With the sequential pixel flow, only one data path is needed since all the pixels are multiplexed into a single stream. However, the data transfer rates can become quite large. For a high resolution 1024 × 1024 image, the data transfer rate is approximately 30M pixels/second.

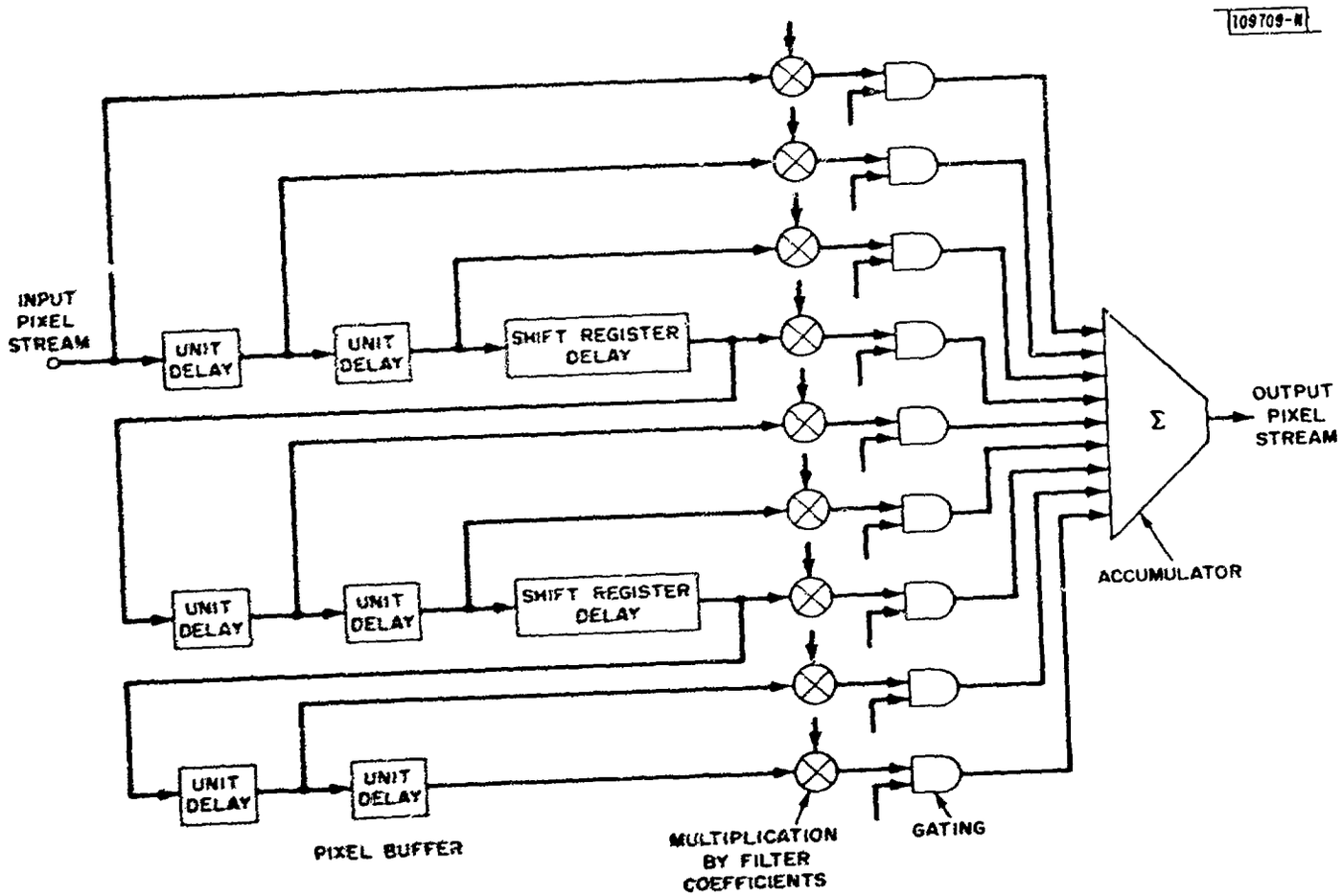


Fig. 21. 3 x 3 convolution with sequential pixel flow.

The "parallel row" image flow offers a compromise between the frame-by-frame and the sequential pixel image flows. The parallel row image flow, as the name implies, transfers rows of image data in parallel, but the pixels within each row are transferred serially. This serves to reduce the number of parallel data paths compared to the frame-by-frame method, and yet it does not require the high transfer rates of the sequential pixel method. Unfortunately, the number of parallel data paths required may still be large for typical image data. The parallel row image flow seems potentially well suited to sensor systems consisting of a linear array of detectors; each row of the image corresponds to data from one of the detectors.

5.2 An Architecture for Iterative Filters Using Sequential Pixel Image Flow

As we saw in Fig. 20, the iterative filter contains two 2-D FIR convolutions, so let us begin by examining the structure for convolution with a sequential pixel image flow. The approach is straightforward; basically, it consists of a pixel buffer which allows simultaneous access to all the input pixels needed to compute a particular output pixel. Figure 21 shows the implementation of a 3×3 FIR filter on a sequential pixel stream. The unit delays in conjunction with the shift register delays form the necessary pixel buffer. The length of the shift register is two less than the length of an image row, so that a shift register delay plus two unit delays will buffer an entire row of pixels. The appropriate nine pixels are multiplied by the FIR filter coefficients and then summed to form a single output pixel. The gating logic shown in Fig. 21 allows one or more of the product terms to be zeroed out; this is necessary to handle edge pixels in the images and the boundary between images.

The convolution operation shown in Fig. 21 may now be used in the implementation of the iterative filter. In effect, Fig. 21 is inserted into the boxes labeled $A(\omega_1, \omega_2)$ and $C(\omega_1, \omega_2)$ in Fig. 20. The frame storage buffer in Fig. 20 can be implemented with a shift register whose length is equal to the number of pixels in an image. It may even be possible to combine the two filter output accumulators with the summing node in Fig. 20 for more efficient operation.

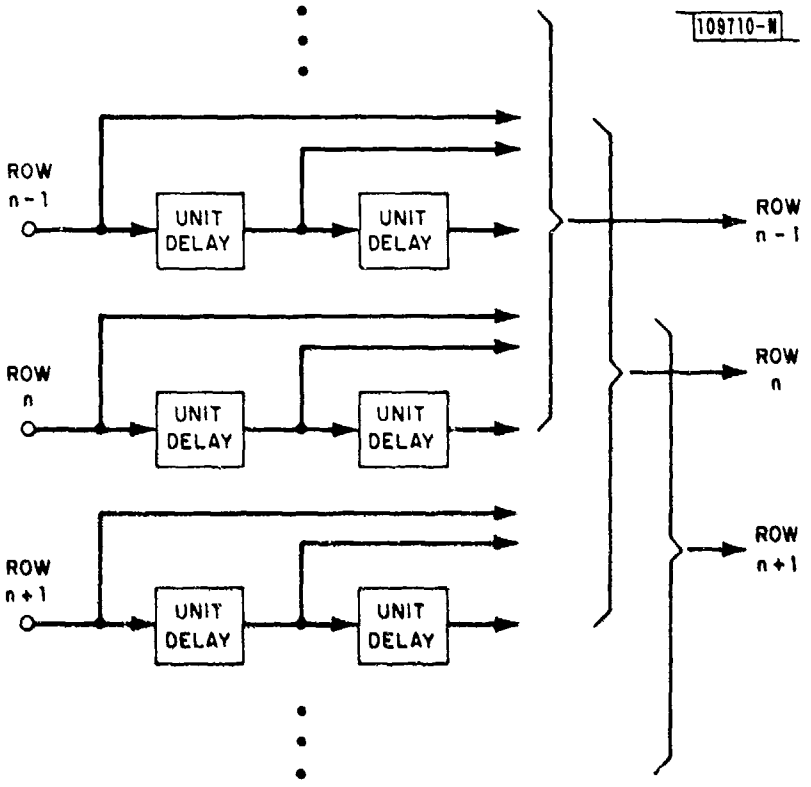


Fig. 22. Structure for a row-parallel convolution. (Braces represent coefficient multiplication, accumulation, and gating.)

There are some flexibility problems with this approach. The size of the pixel buffer in Fig. 21 depends on the size of the FIR filter being implemented, as does the number of coefficient multipliers, control gates, and accumulator inputs. A structure designed to implement convolutions of a fixed size, say 3×3, would not be able to implement larger-size convolutions. On the other hand, a processor designed to support an 11×11 convolution would not be making full use of its computational potential when implementing any smaller-size convolutions. Furthermore, the sizes of the frame storage buffer in Fig. 20 and the gating logic signals in Fig. 21 depend on the number of pixels per image.

The exact length of the frame storage buffer also depends on the assumed regions of support for the impulse responses $a(n_1, n_2)$ and $c(n_1, n_2)$. If $a(n_1, n_2)$ and $c(n_1, n_2)$ are 3×3 with first quadrant support, then the length of the frame buffer is equal to the number of pixels in an image. However, if $a(n_1, n_2)$ and $c(n_1, n_2)$ are 3×3 but centered on the origin, the frame buffer length must be shortened by one row plus one pixel to ensure that the forward pixel stream and the feedback pixel stream are properly synchronized. This detail further reduces the flexibility of this structure for implementing a variety of iterative filters on a variety of image sizes.

5.3 Row Parallel Architectures

The structure for implementing an iterative filter with a row parallel image flow is quite similar in style to the sequential pixel structure discussed above. However, there are some differences in implementation. For example, there is no longer a need for the shift register delays in Fig. 21 since input pixels are available in parallel. Also, the generation of the output pixels in parallel requires the replication of the coefficient multipliers and the accumulator in Fig. 21. Thus, the corresponding structure for the implementation of an FIR filter with row parallel image flow takes the form shown in Fig. 22. The brackets represent the multiplication by filter coefficients, gating, and accumulation functions shown explicitly in Fig. 21. The brackets also serve to indicate the inter-row communication which is necessary to implement a finite-extent convolution.

The structure of Fig. 22 may be substituted for the boxes labeled $A(\omega_1, \omega_2)$ and $C(\omega_1, \omega_2)$ in Fig. 20 to implement an iterative filter. Because of the row parallel image flow, the number of data paths in the iterative filter is equal to the number of rows. This number may be prohibitively large for many applications; pixels may have to be multiplexed to share a smaller number of data paths. This, of course, leads us back in the direction of the sequential pixel image flow.

The row parallel structure also suffers from the inflexibilities described in Sec. 5.2. The number of unit delays, the amount of inter-row communications, and the number of coefficient multipliers depend on the size of the convolutions being implemented. The size of the frame buffer in Fig. 20 depends on the image size as before.

We shall just mention the frame-by-frame image flow in passing, since the number of parallel data paths required, equal to the number of pixels in an image, is impossibly large for current technology. Each output pixel in a given frame is computed in parallel with all the others. For each convolution, this requires a set of coefficient multipliers and accumulators for each pixel. Reducing the number of multipliers and accumulators by multiplexing essentially leads us back toward the row parallel and sequential pixel image flows.

5.4 Multiprocessor Architectures and Algorithm Partitioning

The architectures discussed in Sec. 5.2 and 5.3 were derived from the block diagram of Fig. 20. They seem to possess a certain inflexibility, or hardwired characteristic, that potentially limits their usefulness, although a clever designer may be able to introduce more flexibility into the structure by using random access buffers and microprocessor-controlled multiplexing of the multiplication and accumulation hardware.

Alternatively, we can try a software-oriented approach to the problem of implementing an iterative filter. A single processor with an appropriate amount of random access memory can be straightforwardly programmed to cycle through the necessary computations to produce the output image frames one pixel at a time. Now suppose we have a number of processors at our disposal. The problem becomes one of partitioning the image processing

algorithm, in this case the implementation of an iterative filter, so that each processor is kept busy and is working as independently as possible.

Realistically, the number of processors will be relatively small, say 8. We can partition the image into strips or blocks and assign to each processor the responsibility for computing the output pixels in a particular block. If the size of the block is large compared to the extent of the FIR impulse responses used in the iterative filter, each processor can run independently, as if it were a single processor working on a smaller image, to compute the necessary convolutions on each iteration. Care must be taken where the blocks abut; some inter-processor communication is necessary, either to transfer partially computed output pixels or extra input pixels to neighboring processors.

If one of the processors breaks down, then one block in each output image frame will be garbled or missing. (This block could be moved around from one frame to the next by reassigning processors.) Consequently, this partitioning of the problem does not lead to a very robust implementation. As an alternative, we could subsample the input image frame to obtain several smaller, lower-resolution subimages. A processor could be assigned to process each subimage, and the full output image could be constructed by appropriately interlacing the processed subimages. For example, Fig. 23 shows how an 8×8 image may be partitioned among p = 8 processors. Now each processor has only one-eighth of the input pixels to handle but, for a finite-extent convolution, it must generate a term to contribute to each output pixel in the full image. Each term is roughly one-eighth as complicated as the full sum of products needed to compute a single output pixel, so the amount of computation performed by each

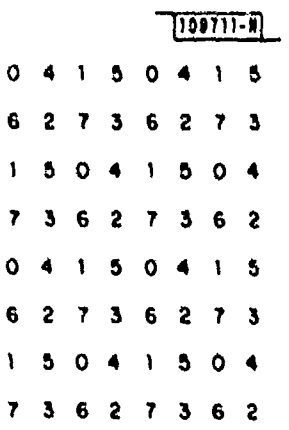


Fig. 23. Partitioning the pixels in an image among 8 processors. The numbers indicate the processor (p0 to p7) to which each pixel is assigned.

processor is essentially the same as if it were computing an output subimage independently of the other processors.

This notion can be made a little more concrete with the example of a 3×3 convolution. The output pixel at (n_1, n_2) is given by

$$\begin{aligned} w(n_1, n_2) = {} & a(-1,-1)\, x(n_1 + 1, n_2 + 2) + a(-1, 0)\, x(n_1 + 1, n_2) \\ & + a(-1, 1)\, x(n_1 + 1, n_2 - 1) + a(0,-1)\, x(n_1, n_2 + 1) \\ & + a(0, 0)\, x(n_1, n_2) + a(0, 1)\, x(n_1, n_2 - 1) \\ & + a(1,-1)\, x(n_1 - 1, n_2 + 1) + a(1, 0)\, x(n_1 - 1, n_2) \\ & + a(1, 1)\, x(n_1 - 1, n_2 - 1) \quad . \end{aligned} \tag{32}$$

If the input image is partitioned as in Fig. 23, then processor p$\emptyset$ must compute the following terms and send them to the other processors:

Source Processor	Destination Processor	Term	For Output Pixel
p$\emptyset$	p$\emptyset$	$a(0, 0)\, x(n_1, n_2)$	$w(n_1, n_2)$
	p1	–	–
	p2	$a(1,-1)\, x(n_1 - 2, n_2 + 2) + a(-1, 1)\, x(n_1, n_2)$	$w(n_1 - 1, n_2 + 1)$
	p3	$a(1, 1)\, x(n_1, n_2) + a(-1,-1)\, x(n_1 + 2, n_2 + 2)$	$w(n_1 + 1, n_2 + 1)$
	p4	$a(1, 0)\, x(n_1, n_2)$	$w(n_1 + 1, n_2)$
	p5	$a(-1, 0)\, x(n_1, n_2)$	$w(n_1 - 1, n_2)$
	p6	$a(0,-1)\, x(n_1, n_2)$	$w(n_1, n_2 - 1)$
	p7	$a(0, 1)\, x(n_1, n_2)$	$w(n_1, n_2 + 1)$

Similarly, processors p1 to p7 must deliver partial results to all 8 processors. Then the partial results can be accumulated to generate the output pixels assigned to each processor.

Let us look briefly at the amount of memory required. We shall assume that image frames consist of N^2 pixels and there are P processors. Each processor needs an N^2/P-pixel input buffer and an N^2/P-pixel output buffer. In addition, each processor must have $N^2(P-1)/P$ storage locations for saving the partial results which will be sent to the other $(P-1)$ processors. Consequently, the total amount of storage required for all processors is $N^2(P+1)$ pixels.

When the processors are done computing the partial outputs, they need to send and receive data to allow them to finish computing the output pixels. We need to postulate an interprocessor switch to accomplish this communication. The p$\emptyset$ output is coupled to the p1 input while the p1 output is coupled to the p2 input, and so on. Then p$\emptyset$ communicates with p2, p1 with p3, and so on. After $P-1$ settings of the switch, the final output pixels can be computed. The total number of pixels transmitted over the interprocessor switch is $N^2(P-1)$, but because of the parallelism only $N^2(P-1)/P$ transfer cycles are used.

The amount of memory used by each processor to buffer the intermediate results can be reduced if the intermediate results can be transmitted shortly after they have been computed. Obviously, the number of pixels transferred and the number of transfer cycles needed do not change; more frequent transfers of shorter data blocks occur.

5.5 Conclusions

The ideas outlined herein should be regarded as preliminary. Nevertheless, some conclusions can be drawn. First, the development of a structure for implementing iterative filters based on the block diagram in Fig. 20 seems to lead to rather inflexible architectures. Second, for particular applications, the type of image flow may depend on the type of sensor (flying spot vs linear array vs lens). Multiplexors and demultiplexors may be used to translate from one image flow to another, of course, but processing architectures may have to be sensitive to the differences in image flows in a real-time application.

At the moment, there seem to be two interesting methods for partitioning the filtering operation which deserve closer scrutiny. The first involves assigning each processor to compute the output pixels in a particular block of the output image frame; the second assigns a subsampled image to each processor so that the effects of a malfunctioning processor will be more diffuse and perhaps less deleterious.

REFERENCES

1. RADC Multi-Dimensional Signal-Processing Research Program, Semiannual Technical Summary, Lincoln Laboratory, M.I.T. (30 September 1980).
2. C.W. Therrien, "Linear Filtering Models for Terrain Image Segmentation," Technical Report 552, Lincoln Laboratory, M.I.T. (4 February 1981), DTIC AD-A099034.
3. O.R. Mitchell, E. J. Delp, and P. L. Chen, "Filtering to Remove Cloud Cover in Satellite Imagery," IEEE Trans. on Geosci. Electron. GE-15, No. 3, 137-141 (July 1977).
4. A. V. Oppenheim, R.W. Schafer, and T.G. Stockham, "Nonlinear Filtering of Multiplied and Convolved Signals," Proc. IEEE 56, 1264-1291 (August 1968), DDC AD-680159.
5. A. Gilkes, "Photograph Enhancement by Adaptive Digital Unsharp Masking," M. S. Thesis, Massachusetts Institute of Technology, Electrical Engineering and Computer Science Department, 1974.
6. T. Peli and J. S. Lim, "Adaptive Filtering for Image Enhancement," Proc. ICASSP '81, p. 1117.
7. W. F. Schreiber, "Image Processing for Quality Improvement," Proc. IEEE 66, No. 12, 1640-1651 (December 1978).
8. NASA Image Filtering Program Semiannual Technical Summary, Lincoln Laboratory, M.I.T. (31 December 1980).

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-81-94	2. GOVT ACCESSION NO. AD-A108641	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Multi-Dimensional Signal-Processing Research Program		5. TYPE OF REPORT & PERIOD COVERED Semiannual Technical Summary 1 October 1980 – 31 March 1981 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dan E. Dudgeon		8. CONTRACT OR GRANT NUMBER(s) F19628-80-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Element No. 62702F Project No. 4594
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center Griffiss AFB, NY 13440		12. REPORT DATE 31 March 1981 13. NUMBER OF PAGES 68
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Electronic Systems Division Hanscom AFB Bedford, MA 01731		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

image classification
iterative image restoration
image segmentation
iterative filtering

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This Semiannual Technical Summary covers the period 1 October 1980 through 31 March 1981. It describes the significant results of the Lincoln Laboratory Multi-Dimensional Signal-Processing Research Program, sponsored by the Rome Air Development Center, in the areas of image segmentation and classification, adaptive contrast enhancement, and iterative implementations for multi-dimensional digital filters.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)